# ● PRINTER RUSH ●
(PTO ASSISTANCE)

CA

| | | | | | |
|---|---|---|---|---|---|
| Application : | 09/587076 | Examiner : | Corrielus | GAU : | 2162 |
| From: | PAP | Location: | [✓ IDC] FMF FDC | Date: | 5/18/05 |

Tracking #: 06075619 Week Date: 2/7/05

| DOC CODE | DOC DATE | MISCELLANEOUS |
|---|---|---|
| ☐ 1449 | | ☐ Continuing Data |
| ☐ IDS | | ☐ Foreign Priority |
| ☐ CLM | | ☐ Document Legibility |
| ☐ IIFW | | ☐ Fees |
| ☐ SRFW | | ☐ Other |
| ☐ DRW | | |
| ☐ OATH | | |
| ☐ 312 | | |
| ☑ SPEC | 6/2/2000 | |

[RUSH] **MESSAGE:** ① Please supply missing Ser. No's. on page 24, lines 4, 7, and 10; page 45, lines 23 and 26; and page 46, line 2.

② Page 44, last sentence on page does not end with a period.

Thank you

[XRUSH] **RESPONSE:** [illegible]

**INITIALS:** [illegible]

NOTE: This form will be included as part of the official USPTO record, with the Response document coded as XRUSH.

REV 10/04

The CNSCS may be embodied as a self-contained message-passing system that may operate among similar small systems, and may be bridged to a complete Jini federation using a bridging server. Examples of such a CNSCS is described in U.S. Provisional Patent Application No. 60/209430 to Slaughter, Saulpaugh, Traversat, Abdelaziz, Duigou, Joy, and Pouyoul, titled "DISTRIBUTED COMPUTING ENVIRONMENT", filed June 2, 2000, which is hereby fully incorporated by reference in its entirety, and in U.S. Provisional Patent Application No. 60/209430 to Slaughter, Saulpaugh, Traversat, Abdelaziz, Duigou, Joy, and Pouyoul, titled "DISTRIBUTED COMPUTING ENVIRONMENT", filed June 2, 2000, which is hereby fully incorporated by reference in its entirety, and in U.S. Provisional Patent Application No. 60/209430 to Slaughter, Saulpaugh, Traversat, Abdelaziz, Duigou, Joy, and Pouyoul, titled "DISTRIBUTED COMPUTING ENVIRONMENT", filed June 2, 2000, which is hereby fully incorporated by reference in its entirety.

K
5/24/05

CNSCS clients are typically small footprint devices that may include small display screens and keyboards. CNSCS clients may be mobile or non-mobile devices. Examples of mobile CNSCS clients may include, but are not limited to: cell phones, palmtop computers, notebook computers, Personal Digital Assistants (PDAs), desktop computers, and printers. An example of a non-mobile CNSCS client may be a light switch comprising a simple chip capable of receiving a simple set of commands (on/off) and of transmitting a simple set of status messages (on/off status).

A CNSCS client may include core CNSCS software and one or more client applications. A CNSCS client may connect to a "fixed" network through a variety of paths. Examples of connection methods may include, but are not limited to: wireless connection (cell phones, Wireless Access Protocol (WAP)), infrared (IrDA), Ethernet, and phone/modem. CNSCS clients may connect to a network through gateways. The gateways provide the client devices with access to CNSCS servers on the network. A gateway may include a proxy CNSCS server. When connected, a CNSCS client "finds" a proximity network on which the client can run one or more applications from the network. One or more of the applications may only be available on the particular one of the devices may not be connected to a network. In other embodiments, non-pure Java applications and/or non-Java applications from one machine to another on a network or between devices when at least one of the devices may not be connected to a network. In order to handle the problem of migrating the external state of an application, migratable applications may use a Network Service Connection System such as Jini or a Compact Network Service Connection System (CNSCS) for accessing resources external to the applications, referred to as services. Services may be local (on the device within which the application is running) or remote (on other devices connected to the device via the network). Local services may include system resources on the device within which the application is running. These local or remote services may be leased by an application using an NSCS or CNSCS. Thus, in one embodiment, the external state of the application may be represented by one or more leases to local and/or remote services, including system resources. Other embodiments may use other methods for accessing external resources that allow for the preservation of external state during migration.

In one embodiment, each application on a system is separated from other applications, and is thus migratable separately from other applications. In one embodiment, each application on a system may have an in-memory heap serving as "physical" memory that is being used for the current execution of the application, a virtual heap that may include the entire heap of the application including at least a portion of the runtime environment of the virtual machine, and a persistent heap or store where the virtual heap may be checkpointed. In one embodiment, the virtual heap and the persistent heap may be combined in one memory (the virtual heap may serve as the persistent heap). In another embodiment, the virtual heap may be checkpointed to a separate, distinct persistent heap. The combination of the in-memory heap, the virtual heap, and the persistent store may be referred to as the "virtual persistent heap." In yet another embodiment, there may be sufficient memory available for the in-memory heap so that a virtual heap is not required to run the application; in this embodiment, only an in-memory heap and a persistent heap on the store may be present for an application.

K
5/24/05

One embodiment of a method for migrating an application may include:

- Checkpointing the application to its persistent heap. In addition, any current leases to external services and/or resources may be expired.
- Packaging the persistent state of the application in the persistent heap and sending the persistent heap for the application to the node where the application is to migrate. In one embodiment, a transaction mechanism is used, where the application's entire persistent state may be copied atomically as a "transaction" and committed as having migrated on both the sending and receiving nodes.
- Reconstituting the state of the application into a new persistent heap (may be a virtual persistent heap) on the node where the application migrated.
- Re-establishing leases to external services and/or resources for the application.
- The application resuming execution in the persistent heap on the node where it migrated.

In one embodiment, since processes that migrate away from a node may migrate back after minor state changes on the node where they migrated (e.g. updated a page of a document), a versioning mechanism may be used whereby nodes where an application once lived may cache a previous state, and thus may avoid sending over the network a state that hasn't changed.

Information on the current leases for the application may also be packaged and sent to the new node where the application is to migrate. The information may be used in re-establishing the leases on the new node. In one embodiment, the lease information may be maintained in a gate structure. Examples of gate structures for a CNSCS is described in U.S. Provisional Patent Application No. 60/209430 to Slaughter, Saulpaugh, Traversat, Abdelaziz, Duigou, Joy, and Pouyoul, titled "DISTRIBUTED COMPUTING ENVIRONMENT", filed June 2, 2000, which was previously fully incorporated by reference in its entirety, and in U.S. Provisional Patent Application No. 60/209430 to Slaughter, Saulpaugh, Traversat, Abdelaziz, Duigou, Joy, and Pouyoul, titled "DISTRIBUTED COMPUTING ENVIRONMENT", filed June 2, 2000, which was

TC
5/24/05

TR
5/24/05

previously fully incorporated by reference in its entirety, and in U.S. Provisional Patent Application No. 60/209430 to Slaughter, Saulpaugh, Traversat, Abdelaziz, Duigou, Joy, and Pouyoul, titled "DISTRIBUTED COMPUTING ENVIRONMENT", filed June 2, 2000, which was previously fully incorporated by reference in its entirety.

In addition, a user interface (UI) may be provided to manage application checkpoints. Functions the UI may allow the user to perform may include, but are not limited to, the following:

- Browse the store.
- Select an application checkpoint to restart.
- Suspend the current application.
- Remove an application checkpoint.

Figure 5a is a block diagram illustrating an embodiment of an application migration process where the original application 104a and the migrated application 104b may use the same virtual heap 110 in persistent store 120. In Figure 5a, the in-memory heap 108 for application 104a executing on client system 100 is checkpointed to persistent store 120. The checkpointing may be performed as an atomic transaction. The store checkpoint may include one or more of the following states that may be made permanent to the store:

- All dirty user pages since the beginning of the transaction.
- All dirty system pages since the beginning of the transaction.
- The current state of non-heap (for example, virtual machine) internal structures (thread contexts, pointer to main structure in the heap such as classes, constant pool, etc.).

Any current leases to external services (for example, services leased via an NSCS such as Jini or a CNSCS) may be expired. In one embodiment, expiration of current leases may be required prior to migration. In one embodiment, expiration of current leases is not required before checkpointing the application.